In the Claims

Please amend the claims as follows:

1. (Currently Amended) A processor comprising:

a first memory configured to store a sequence of instructions representing essential code;

a second memory configured to store sequences of instructions representing non-essential code;

a conjugate mapping table configured with a plurality of triggers to ~~specify respectively~~ relate different sequences of the non-essential code to ~~be executed from the second memory; and~~ respective contingent conditions that specify predefined attributes of the essential code;

a single microarchitecture structure configured to execute instructions both from the essential code and from the ~~nonessential~~ non-essential code, the processor being coupled to both the first and second memories to process code from the first memory, and to process code from the second memory in response to the triggers.

2. (Currently Amended) The processor of claim 1 wherein the first ~~pipeline~~ memory is coupled to a first instruction cache configured to cache instructions that determine the logical correctness of a program.

3. (Currently Amended) ~~The processor of claim 2~~ A processor comprising:

a first memory configured to store a sequence of instructions representing essential code;

a second memory configured to store sequences of instructions representing non-essential code;

a conjugate mapping table configured with a plurality of triggers to specify respectively different sequences of the non-essential code to be executed from the second memory; and

a single microarchitecture structure configured to execute instructions both from the essential code and from the non-essential code, the processor being coupled to both the first and second memories to process code from the first memory, and to process code from the second memory in response to the triggers.

wherein the first memory is coupled to a first instruction cache configured to cache instructions that determine the logical correctness of a program,

wherein the second ~~pipeline~~ memory is coupled to a second instruction cache configured to cache instructions that provide hints for the execution of the instructions that determine the logical correctness of the program.

4. (Currently Amended) The processor of claim 1 wherein the first ~~pipeline~~ memory is coupled to registers that store a microarchitectural state, and wherein the conjugate mapping table is responsive to the microarchitectural state.

5. (Currently Amended) The processor of claim 4 further comprising:

a first instruction cache coupled to the first ~~pipeline; and~~ memory;

a second instruction cache coupled between the conjugate mapping table and the second ~~pipeline~~ memory.

6. (Canceled)

7. (Original) The processor of claim 1 wherein the conjugate mapping table comprises a plurality of records, each of the plurality of records being configured to map a trigger to a non-essential code sequence.

8. (Original) The processor of claim 7 wherein the trigger comprises an atomic value, such that the conjugate mapping table is configured to specify the non-essential code sequence when the atomic value is satisfied.

9. (Original) The processor of claim 7 wherein the trigger comprises a vector value, such that the conjugate mapping table is configured to specify the non-essential code sequence when the vector value is satisfied.

10 (Canceled)

11. (Previously Presented) The processor of claim 1 wherein the microarchitectural structure includes a register bank.

12-30. (Canceled)

31. (Currently Amended) The processor of claim 1 further comprising a dynamic code analyzer to generate non-essential code from the essential code in the first ~~pipeline~~ memory.

32. (Currently Amended) The processor of claim 1 ~~where the~~ further comprising a dynamic code analyzer ~~creates~~ directed acyclic graph trace representations of at least some of the essential code from the first ~~pipeline~~ memory.

33. (Currently Amended) A method comprising:

loading a first stream of instructions containing essential code into a first memory;

loading a second stream of instructions containing non-essential code into a separate second memory;

storing a mapping table relating a plurality of triggers to respective ones of a plurality of different sequences of the non-essential code, where each trigger includes at least one of a plurality of different conditions resulting from execution of the essential code;

executing instructions from the essential code from the first memory in a microarchitecture structure until detecting an occurrence one of the triggers; ~~and~~

thereafter, executing instructions from one of the non-essential code sequences in the same microarchitecture structure, the one sequence being specified by the at least one condition of the one trigger.

34. (Previously Presented) The method of claim 33 wherein the first and second memories are a first and second pipelines.

35. (Previously Presented) The method of claim 33 wherein the first and second memories are caches.

36. (Previously Presented) The method of claim 35 wherein the first and second memories are logically separate.

37. (Previously Presented) The method of claim 35 wherein the first and second memories are physically separate.

38. (Currently Amended) ~~The method of claim 33~~ A method comprising:

loading a first stream of instructions containing essential code into a first memory;

loading a second stream of instructions containing non-essential code into a separate second memory;

storing a mapping table relating a plurality of triggers to respective ones of a plurality of different sequences of the non-essential code;

executing instructions from the essential code from the first memory in a microarchitecture structure until detecting an occurrence one of the triggers;

thereafter, executing instructions from one of the non-essential code sequences in the same microarchitecture structure, the one sequence being specified by the one trigger,

wherein the non-essential code includes hint code generated by a compiler from the essential code.

39. (Previously Presented) The method of claim 33 wherein the non-essential code includes sequences to perform instruction set virtualization.

40. (Previously Presented) The method of claim39 wherein the included sequences perform respectively multiple ones of the functions.

41. (Previously Presented) The method of claim 33 wherein certain of the essential code is stored in the second memory.

42. (Previously Presented) The method of claim 41 wherein the certain essential code includes sequences for virtualization.

43. (Previously Presented) The method of claim 42 wherein at least one of the sequences virtualizes individual instructions.

44. (Previously Presented) The method of claim 33 where the triggers are instruction attributes.

45. (Previously Presented) The method of claim 44 where the instructions attributes include opcodes, locations, and/or operands.

46. (Previously Presented) The method of claim 65 wherein the data attributes include values and/or locations.

47. (Previously Presented) The method of claim 66 wherein the state attributes include architectural and/or microarchitectural states.

48. (Previously Presented) The method of claim 67 wherein the event attributes include interrupts, exceptions, and/or processor state register values.

49. (Currently Amended) A system comprising:

storage to store a single library having

a first part containing instructions representing essential code for executing a particular program and

a second part containing sequences of instructions representing ~~nonessential~~ non-essential code associated with the same particular program, wherein the non-essential code includes code generated from the essential code;

a processor including

first and second memories to store the essential and non-essential code respectively,

a mapping table to relate a plurality of triggers to a plurality of sequences of the non-essential code,

a single microarchitecture structure coupled to the memories to execute at least some of both the essential code and the sequences of ~~nonessential~~ non-essential code in response to their respective triggers.

50. (Previously Presented) The system of claim 49 wherein the memory includes at least one cache.

51. (Previously Presented) The system of claim 49 wherein the memory includes one or more of a hard disk, a floppy disk, RAM, ROM, a flash memory, and/or a medium readable by a machine.

52. (Previously Presented) The system of claim 49 where the memory stores the essential and non-essential code in separate sections of a file.

53. (Previously Presented) The system of claim 52 where the essential and nonessential code reside in a static file.

54. (Previously Presented) The system of claim 52 where the nonessential code resides at least partly in a run-time library.

55. (Currently Amended) The system of claim 1 where the single microarchitecture structure is configured to execute all instructions from both the essential and the ~~nonessential~~ non-essential code.

56. (Currently Amended) The system of claim 1 where the second ~~pipeline~~ memory further contains some code that is not ~~nonessential~~ non-essential.

57. (Previously Presented) The method of claim 33 wherein the non-essential code includes sequences to perform interrupt or exception processing.

58. (Previously Presented) The method of claim 33 wherein the non-essential code includes sequences to perform speculative execution.

59. (Previously Presented) The method of claim 33 wherein the non-essential code includes sequences to perform security checking or sandboxing.

60. (Previously Presented) The method of claim 33 wherein the non-essential code includes sequences to test the microarchitecture.

61. (Currently Amended) ~~The method of claim 33~~ A method comprising:

loading a first stream of instructions containing essential code into a first memory;

loading a second stream of instructions containing non-essential code into a separate second memory;

storing a mapping table relating a plurality of triggers to respective ones of a plurality of different sequences of the non-essential code;

executing instructions from the essential code from the first memory in a microarchitecture structure until detecting an occurrence one of the triggers;

thereafter, executing instructions from one of the non-essential code sequences in the same microarchitecture structure, the one sequence being specified by the one trigger,

wherein the non-essential code includes sequences to prefetch essential code into the first memory.

62. (Previously Presented) The method of claim 42 wherein at least one of the sequences virtualizes blocks of instructions.

63. (Previously Presented) The method of claim 42 wherein at least one of the sequences virtualizes sets of registers,.

64. (Previously Presented) The method of claim 42 wherein at least one of the sequences virtualizes processor hardware resources.

65. (Previously Presented) The method of claim 33 where the triggers are data attributes.

66. (Previously Presented) The method of claim 33 where the triggers are state attributes.

67. (Previously Presented) The method of claim 33 where the triggers are event attributes.